Trainable Videorealistic Speech Animation

by

Tony Farid Ezzat

S.B, M.Eng, Massachusetts Institute of Technology, 1996

Submitted to the Department of Electrical Engineering and Computer
Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2002

Author ....
Department of Electrical Engineering and Computer Science
June 6, 2002

Certified by...
Tomaso Poggio
Whitaker Professor
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

	Form Approved
# Report Documentation Page	OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **JUN 2002**	2. REPORT TYPE	3. DATES COVERED **00-06-2002 to 00-06-2002**
4. TITLE AND SUBTITLE **Trainable Videorealistic Speech Animation**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Massachusetts Institute of Technology,Department of Electrical Engineering and Computer Science,Cambridge,MA,02139**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES **The original document contains color images.**

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES **58**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Trainable Videorealistic Speech Animation

by

Tony Farid Ezzat

S.B, M.Eng, Massachusetts Institute of Technology, 1996

Abstract

I describe how to create with machine learning techniques a generative, videorealistic, speech animation module. A human subject is first recorded using a videocamera as he/she utters a pre-determined speech corpus. After processing the corpus automatically, a visual speech module is learned from the data that is capable of synthesizing the human subject's mouth uttering entirely novel utterances that were not recorded in the original video. The synthesized utterance is re-composited onto a background sequence which contains natural head and eye movement. The final output is videorealistic in the sense that it looks like a video camera recording of the subject. At run time, the input to the system can be either real audio sequences or synthetic audio produced by a text-to-speech system, as long as they have been phonetically aligned.

The two key contributions of this work are

- a variant of the *multidimensional morphable model* (MMM) [4] [26] [25] to synthesize new, previously unseen mouth configurations from a small set of mouth image prototypes,

- a *trajectory synthesis technique* based on regularization, which is automatically trained from the recorded video corpus, and which is capable of synthesizing trajectories in MMM space corresponding to any desired utterance.

Results are presented on a series of numerical and psychophysical experiments designed to evaluate the synthetic animations.

Thesis Supervisor: Tomaso Poggio
Title: Whitaker Professor

Acknowledgements

First and foremostly, I would like to thank my advisor Tommy Poggio for hiring me, giving me the freedom and support to pursue this project in his lab, and providing such an exciting work environment at CBCL.

Also I would like to thank Gadi Geiger for collaborating with me on the psychophysics portion of this thesis.

At CBCL, I had the pleasure of stimulating interaction with a large number of smart individuals from whom I have learned a lot: Vinay Kumar, Massimiliano Pontil, Theos Evgeniou, Federico Girosi, Sayan Mukherjee, Luis Perez Breva, Martin Szummer, Ryan Rifkin, Adlar Kim, Berndt Heisele, Osamu Yoshimi, Constantine Papageorgiou, Purdy Ho, Sanmay Das, and Nicholas Chan. I would especially like to thank Marypat Fitzgerald and Casey Johnson for all their help.

At the AI Lab, I also had the privilege of interacting with a number of graduate students who would provide the groundwork for this work. In this regard, I would like to thank and acknowledge David Beymer, Mike Jones, Steve Lines, Roberto Brunelli, Volker Blanz, and Thomas Vetter.

Carolos Livadas was my closest MIT friend throughout this entire trip from MIT undergrad all the way up to graduation, and I acknowledge his friendship, loyalty, hard work, and putting up with all that agony with Prosopa.

I would also like to thank my parents for their constant support, as well as the support of my brother Chris and sister-in-law Angela.

Due to the events surrounding the Boston Globe article which appeared on May 15, 2002 detailing the repercussions of this work, I have chosen to submit a condensed form of this thesis. A full and complete technical report on this work will be forthcoming from me in the latter half of 2002.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Is it possible to record a human subject with a video camera, process the recorded data automatically, and then re-animate that subject uttering entirely novel utterances which were not included in the original corpus? In this work, we present such a technique for achieving videorealistic speech animation.

We choose to focus our efforts in this work on the issues related to the synthesis of novel video, and not on novel audio synthesis. Thus, novel audio needs to be provided as input to our system. This audio can be either real human audio (from the same subject or a different subject), or synthetic audio produced by a text-to-speech system. All that is required by our system is that the audio be phonetically transcribed and aligned. In the case of synthetic audio from TTS systems, this phonetic alignment is readily available from the TTS system itself [7]. In the case of real audio, publicly available phonetic alignment systems [24] may be used.

Our visual speech processing system is composed of two modules: The first module is the *multidimensional morphable model* (MMM), which is capable of morphing between a small set of prototype mouth images to synthesize new, previously unseen mouth configurations. The second component is a *trajectory synthesis* module, which uses regularization [21] [42] to synthesize smooth trajectories in MMM space for any specified utterance. The parameters of the trajectory synthesis module are trained automatically from the recorded corpus using gradient descent learning.

Recording the video corpus takes on the order of 15 minutes. Processing of the
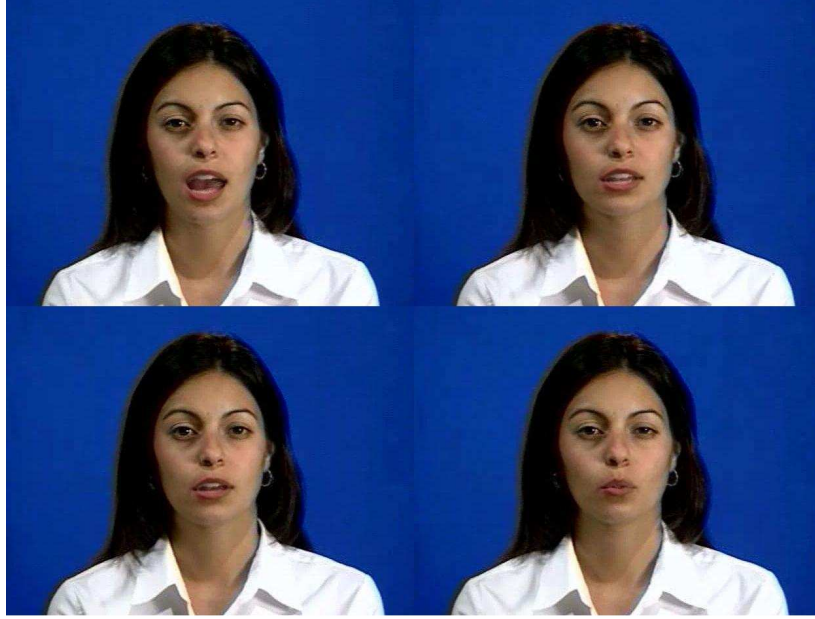
Figure 1-1: Some of the synthetic facial configurations output by our system.

corpus takes on the order of several days, but, apart from the specification of head and eye masks shown in Figure 5-1, is *fully automatic*, requiring no intervention on the part of the user. The final visual speech synthesis module consists of a *small* set of prototype images (46 images in the case presented here) extracted from the recorded corpus and used to synthesize all novel sequences.

Application scenarios for videorealistic speech animation include: user-interface agents for desktops, TVs, or cell-phones; digital actors in movies; virtual avatars in chatrooms; very low bitrate coding schemes (such as MPEG4); and studies of visual speech production and perception. The recorded subjects can be regular people, celebrities, ex-presidents, or infamous terrorists.

In the following section, we begin by first reviewing the relevant prior work and motivating our approach.

Chapter 2

Background

2.1 Facial Modeling

One approach at facial modeling is to model the face using *3D modeling* methods. Parke [33] was one of the earliest to adopt such an approach by creating a polygonal facial model. To increase the visual realism of the underlying facial model, the facial geometry is frequently scanned in using Cyberware laser scanners. Additionally, a texture-map of the face extracted by the Cyberware scanner may be mapped onto the three-dimensional geometry [29]. Guenter [22] demonstrated recent attempts at obtaining 3D face geometry from multiple photographs using photogrammetric techniques. Pighin et al. [35] captured face geometry and textures by fitting a generic face model to a number of photographs. Blanz and Vetter [9] demonstrated how a large database of Cyberware scans may be morphed to obtain face geometry from a single photograph.

An alternative to the 3D modeling approach is to model the talking face using *image-based* techniques, where the talking facial model is constructed using a collection of example images captured of the human subject. These methods have the potential of achieving very high levels of videorealism, and are inspired by the recent success of similar sample-based methods for audio speech synthesis [32].

Image-based facial animation techniques need to solve the *video generation* problem: How does one build a generative model of novel video that is simultaneously

photorealistic, *videorealistic*, and *parsimonious*? *Photorealism* means that the novel generated images exhibit the correct visual structure of the lips, teeth, and tongue. *Videorealism* means that the generated *sequences* exhibit the correct motion, dynamics, and coarticulation effects [16]. *Parsimony* means that the generative model is represented compactly using a few parameters.

2.1.1 Video Rewrite

Bregler, Covell, and Slaney [12] describe an image-based facial animation system called Video Rewrite in which the video generation problem is addressed by breaking down the recorded video corpus into a set of smaller audiovisual basis units. Each one of these short sequences is a *triphone* segment, and a large database with all the acquired triphones is built. A new audiovisual sentence is constructed by *concatenating* the appropriate triphone sequences from the database together. Photorealism in Video Rewrite is addressed by only using recorded sequences to generate the novel video. Videorealism is achieved by using triphone contexts to model coarticulation effects. In order to handle all the possible triphone contexts, however, the system requires a library with tens and possibly hundreds of thousands of subsequences, which seems to be an overly-redundant and non-parsimonious sampling of human lip configurations. Parsimony is thus sacrificed in favor of videorealism.

Essentially, Video Rewrite adopts a decidedly *agnostic* approach to animation: since it does not have the capacity to *generate* novel lip imagery from a few recorded images, it relies on the re-sequencing of a vast amount of original video. Since it does not have the capacity to *model* how the mouth moves, it relies on sampling the dynamics of the mouth using triphone segments.

2.1.2 Multidimensional Morphable Models

The approach used in this work presents another approach to solving the video generation problem which has the capacity to *generate novel video from a small number of examples* as well as the capacity *to model how the mouth moves*. This approach is

14

based on the use of a *multidimensional morphable model* (MMM), which is capable of multdimensional morphing between various lip images to synthesize new, previously unseen lip configurations. MMM's have already been introduced in other works [36] [4] [17] [25] [28] [9] [8]. In this work, we develop an MMM variant and show its utility for facial animation.

MMM's are powerful models of image appearance because they combine the power of *vector space* representations with the realism of *morphing* as a generative image technique. Prototype example images of the mouth are decomposed into pixel *flow* and pixel *appearance* axes that represent *basis vectors* of image variation. These basis vectors are combined in a multidimensional fashion to produce novel, realistic, previously unseen lip configurations.

As such, an MMM is more powerful than other vector space representations of images which do not model pixel flow explicitly. Cosatto and Graf [19], for example, describe an approach which is similar to ours, except that their generative model involved simple pixel blending of images, which fails to produce realistic transitions between mouth configurations.

An MMM is also more powerful than simple 1-dimensional morphing between 2 image end-points [2], as well as techniques such as those of Scott, Kagels, et al. [38] [44] and Ezzat and Poggio [20], which morphed between several visemes in pairwise fashion. By embedding the prototype images in a vector space, an MMM is capable of generating *smooth curves* through lip space which handle complex speech animation effects in a non-ad-hoc manner.

2.2 Speech Animation

Speech animation techniques have traditionally included both keyframing methods and physics-based methods, and have been extended more recently to include machine learning methods. In keyframing, the animator specifies particular key-frames, and the system generates intermediate values [33] [34] [16] [30]. In physics-based methods, the animator relies on the laws of physics to determine the mouth move-

ment, given some initial conditions and a set of forces for all time. This technique, which requires modeling the underlying facial muscles and skin, was demonstrated quite effectively by [43] [29]. Finally, machine learning methods are a new class of animation tools which are trained from recorded data and then used to synthesize new motion. Examples include hidden markov models (HMMs), which were demonstrated effectively for speech animation by [10] [31] [13].

Speech animation needs to solve several problems simultaneously: firstly, the animation needs to have the correct *motion*, in the sense that the appropriate phonemic targets need to be realized by the moving mouth. Secondly, the animation needs to be *smooth*, not exhibiting any unnecessary jerks. Thirdly, it needs to display the correct *dynamics*: plosives such as b and p need to occur fast. Finally, speech animation needs to display the correct *coarticulation effects*, which determine the effects of neighboring phonemes on the current phoneme shape.

In this work, we present a *trajectory synthesis* module to address the issues of synthesizing mouth trajectories with correct motion, smoothness, dynamics, and coarticulation effects. This module maps from an input stream of phonemes (with their respective frame durations) to a *trajectory* of MMM shape-appearance parameters. This trajectory is then fed into the MMM to synthesize the final visual stream that represents the talking face.

Unlike Video Rewrite [12], which relies on an exhaustive sampling of triphone segments to model phonetic contexts, coarticulation effects in our system emerge directly from our speech model. Each phoneme in our model is represented as a localized Gaussian target region in MMM space with a particular position and *covariance*. The covariance of each phoneme acts as a spring whose tension pulls the trajectory towards each phonetic region with a force proportional to observed coarticulation effects in the data.

However, unlike Massaro and Cohen [16] (who also modeled coarticulation using localized Gaussian-like regions), our model of coarticulation is not hand-tuned, but rather trained from the recorded corpus itself using a gradient descent learning procedure. The training process determines the position and shape of the phonetic regions

16

in MMM space in a manner which optimally reconstructs the recorded corpus data.

Chapter 3

System Overview
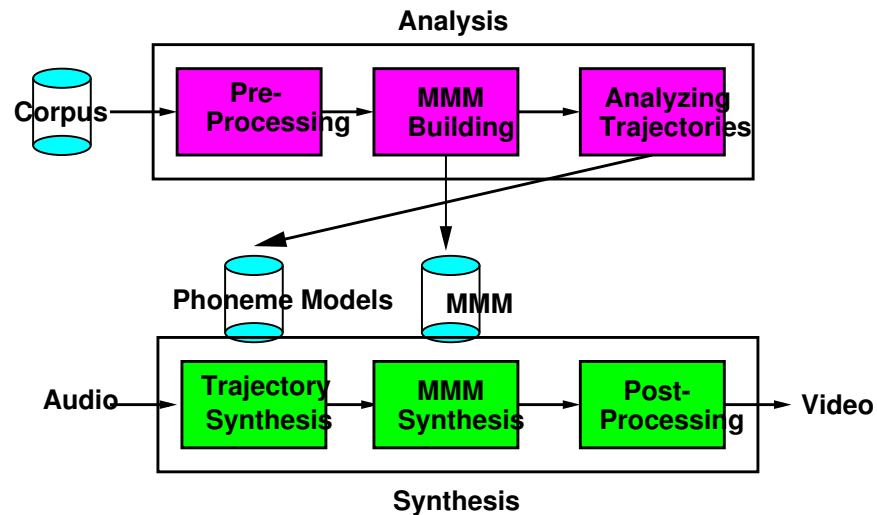
Figure 3-1: An overview of our videorealistic facial animation system.

An overview of our system is shown in Figure 3-1. After recording the corpus (Section 4), *analysis* is performed to produce the final visual speech module. Analysis itself consists of three sub-steps: First, the corpus is pre-processed (Section 5) to align the audio and normalize the images to remove head movement. Next, the MMM is created from the images in the corpus (Section 6.2). Finally, the corpus sequences are analyzed to produce the phonetic models used by the trajectory synthesis module (Sections 6.4 and 7.2).

Given a novel audio stream that is phonetically aligned, synthesis proceeds in three

steps: First, the trajectory synthesis module is used to synthesize the trajectory in MMM space using the trained phonetic models (Section 7). Secondly, the MMM is used to synthesize the novel visual stream from the trajectory parameters (Section 6.3). Finally, the post-processing stage composites the novel mouth movement onto a background sequence containing natural eye and head movements (Section 8).

Chapter 4

Corpus

An audiovisual corpus of a human subject uttering various utterances was recorded. Recording was performed at a TV studio against a blue "chroma-key" background with a standard Sony analog TV camera. The data was subsequently digitized at a 29.97 fps NTSC frame rate with an image resolution of 640 by 480 and an audio resolution of 44.1KHz. The final sequences were stored as Quicktime sequences compressed using a Sorenson coder. The recorded corpus lasts for 15 minutes, and is composed of approximately 30000 frames.

The recorded corpus consisted of 1-syllable and 2-syllable words, such as ``bed'' and ``dagger''. A total of 152 1-syllable words and 156 2-syllable words were recorded. In addition, the corpus included 105 short sentences, such as ``The statue was closed to tourists Sunday''. The subject was asked to utter all sentences in a neutral expression. In addition, the sentences themselves were designed to elicit no emotions from the subject.

Chapter 5

Pre-Processing

The recorded corpus data needs to be pre-processed in several ways before it may be processed effectively for re-animation.

5.1 Audio Alignment

Firstly, the audio needs to be phonetically aligned in order to be able to associate a phoneme for each image in the corpus. We perform audio alignment on all the recorded sequences using the CMU Sphinx system [24], which is publicly available. Given an audio sequence and an associated text transcript of the speech being uttered, alignment systems use forced Viterbi search to find the optimal start and end of phonemes for the given audio sequence. The alignment task is easier than the speech recognition task because the text of the audio being uttered is known apriori.

5.2 Head Movement Normalization

Secondly, each image in the corpus needs to be *normalized* so that only movement occurring in the entire frame is the mouth movement associated with speech. Although the subject was instructed to keep her head steady during recording, residual head movement nevertheless still exists in the final recorded sequences. Since the head motion is small, we make the simplifying assumption that it can be approxi-
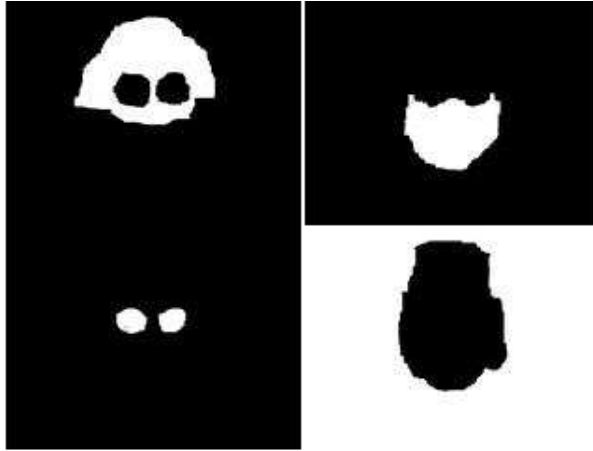
Figure 5-1: The head, mouth, eye, and background masks used in the pre-processing and post-processing steps. Specification of these masks is the only manual step required by this system.

mated as the perspective motion of a plane lying on the surface of the face. Planar perspective deformations [45] have 8 degrees of freedom, and can be inferred using 4 corresponding points between a reference frame and the current frame. We employ optical flow [23] [1] [3] to extract correspondences for 640x480 pixels, and use least squares to solve the overdetermined system of equations to obtain the 8 parameters of the perspective warp. Among the 640x480 correspondences, only those lying within the *head mask* shown in Figure 5-1 are used. Pixels from the background area are not used because they do not exhibit any motion at all, and those from the mouth area exhibit non-rigid motion associated with speech.

After computing the 8 planar perspective parameters, the image is warped towards the reference frame. This is performed for all images in the corpus. After warping the images are cropped to a dimension of 624x420 to eliminate the border artifacts associated with the warp.

The images in the corpus also exhibit residual eye movement and eye blinks which need to be removed. An eye mask is created (see Figure 5-1) which allows just the eyes from a single frame to be pasted onto the rest of the corpus imagery. The eye mask is blurred at the edges to allow a seamless blend between the pasted eyes and

the rest of face.

Chapter 6

Multidimensional Morphable Models

At the heart of our visual speech synthesis approach is the *multidimensional morphable model* representation, which is a *generative model of video* capable of morphing between various lip images to synthesize new, previously unseen lip configurations.

The basic underlying assumption of the MMM is that the complete set of mouth images associated with human speech lies in a low-dimensional space whose axes represent mouth *appearance* variation and mouth *shape* variation. Mouth appearance is represented in the MMM as a set of prototype images extracted from the recorded corpus. Mouth shape is represented in the MMM as a set of optical flow vectors [23] computed automatically from the recorded corpus. In the work presented here, 46 images are extracted and 46 optical flow correspondences are computed. The low-dimensional MMM space is parameterized by shape parameters α and appearance parameters β.

The MMM may be viewed as a "black box" capable of performing two tasks: Firstly, given as input a set of parameters (α, β), the MMM is capable of *synthesizing* an image of the subject's face with that shape-appearance configuration. Synthesis is performed by morphing the various prototype images to produce novel, previously unseen mouth images which correspond to the input parameters (α, β).

Conversely, the MMM can also perform *analysis*: given an input lip image, the

MMM computes shape and appearance parameters (α, β) that represent the position of that input image in MMM space. In this manner, it is possible to *project* the entire recorded corpus onto the constructed MMM, and produce a time series of (α_t, β_t) parameters that represent trajectories of mouth motion in MMM space. We term this operation *analyzing the recorded corpus*.

In the following sections, we describe how a multidimensional morphable model is defined, how it may be acquired automatically from a recorded video corpus, how it may be used for synthesis, and, finally, how such a morphable model may be used for analysis.

6.1 Definition

An MMM consists of a *set of prototype images* $\{I_i\}_{i=1}^N$ that represent the various lip textures that will be encapsulated by the MMM. One image is designated arbitrarily to be the *reference image I_1*.

Additionally, the MMM consists of a set of *prototype flows* $\{C_i\}_{i=1}^N$ that represent the correspondences between the reference image I_1 and the other prototype images in the MMM. The correspondence from the reference image to itself, C_1, is designated to be an empty, zero, flow.

In this work, we choose to represent the correspondence maps using relative displacement vectors:

$$C_i(\mathbf{p}) = \{d_x^i(\mathbf{p}), d_y^i(\mathbf{p})\}. \tag{6.1}$$

A pixel in image I_1 at position $\mathbf{p} = (x, y)$ corresponds to a pixel in image I_i at position $(x + d_x^i(x, y), y + d_y^i(x, y))$.

Previous methods for computing correspondence [2] [38] [27] adopted *feature-based* approaches, in which a set of high-level *shape features* common to both images is specified. When it is done by hand, however, this feature specification process can become quite tedious and complicated, especially in cases when a large amount of imagery is involved. In this work, we make use of *optical flow* [23] [1] [3] algorithms to estimate

25
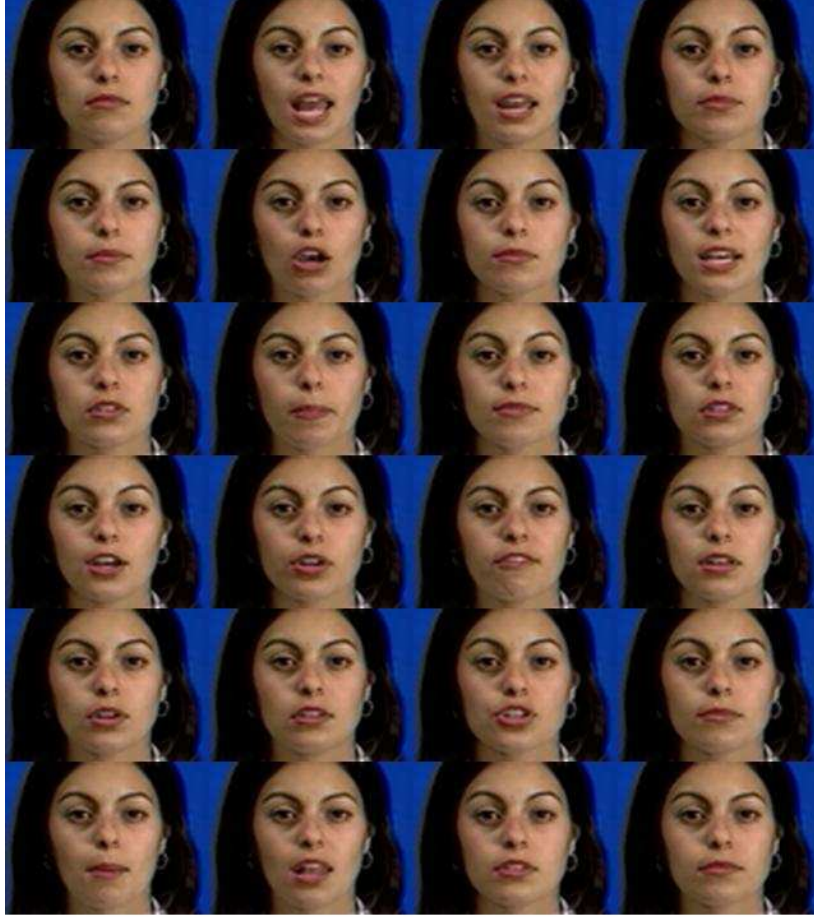
Figure 6-1: 24 of the 46 image prototypes included in the MMM. The reference image is the top left frame.

this motion. This motion is captured as a two-dimensional array of displacement vectors, in the same exact format shown in Equation 6.1. In particular, we utilize the *coarse-to-fine, gradient-based* optical flow algorithms developed by [3]. These algorithms compute the desired flow displacements using the spatial and temporal image derivatives. In addition, they embed the flow estimation procedure in a multiscale pyramidal framework [14], where initial displacement estimates are obtained at coarse resolutions, and then propagated to higher resolution levels of the pyramid.

6.2 Building an MMM

An MMM must be constructed automatically from a recorded corpus of $\{I_j\}_{j=1}^S$ images. The two main tasks involved are to choose the image prototypes $\{I_i\}_{i=1}^N$, and to compute the correspondence $\{C_i\}_{i=1}^N$ between them. We discuss the steps to do this briefly below. Note that the following operations are performed on the entire face region, although they need only be performed on the region around the mouth.

6.2.1 PCA

For the purpose of more efficient processing, principal component analysis (PCA) is first performed on all the images of the recorded video corpus. PCA allows each image in the video corpus to be represented using a set of low-dimensional parameters. This set of low-dimensional parameters may thus be easily loaded into memory and processed efficiently in the subsequent clustering and Dijkstra steps.

Performing PCA using classical autocovariance methods [6], however, usually requires loading all the images and computing a very large autocovariance matrix, which requires a lot of memory. To avoid this, we adopt an on-line PCA method, termed EM-PCA [37] [41], which allows us to perform PCA on the images in the corpus without loading them all into memory. EM-PCA is iterative, requiring several iterations, but is guaranteed to converge in the limit to the same principal components that would be extracted from the classical autocovariance method. The EM-PCA algorithm is typically run in this work for 10 iterations.

Performing EM-PCA produces a set of D 624x472 principal components and a matrix Σ of eigenvalues. In this work, $D = 15$ PCA bases are retained. The images in the video corpus are subsequently projected on the principal components, and each image I_j is represented with a D-dimensional parameter vector p_j.

6.2.2 K-means Clustering

Selection of the prototype images is performed using *k-means clustering* [6]. The algorithm is applied directly on the $\{p_j\}_{j=1}^S$ low dimensional PCA parameters, producing

N cluster centers. Typically the cluster centers extracted by k-means clustering do not coincide with actual image datapoints, so the nearest images in the dataset to the computed cluster centers are chosen to be the final image prototypes $\{I_i\}_{i=1}^N$ for use in our MMM.

It should be noted that k-means clustering requires the use of an internal distance metric with which to compare distances between datapoints and the chosen cluster centers. In our case, since the image parameters are themselves produced by PCA, the appropriate distance metric between two points p_m and p_n is the *Mahalanobis distance metric*:

$$d(p_m, p_n) = (p_m - p_n)^T \Sigma^{-1} (p_m - p_n) \tag{6.2}$$

where Σ is the afore-mentioned matrix of eigenvalues extracted by the EM-PCA procedure.

We selected $N = 46$ image prototypes in this work, which are partly shown in Figure 6-1. The top left image is the reference image I_1. There is nothing magical about our choice of 46 prototypes, which is in keeping with the typical number of visemes other researchers have used [38] [20]. It should be noted, however, that the 46 prototypes have no explicit relationship to visemes, and instead form a simple *basis set* of image textures.

6.2.3 Dijkstra

After the $N = 46$ image prototypes are chosen, the next step in building an MMM is to compute correspondence between the reference image I_1 and all the other prototypes. Although it is in principle possible to compute *direct* optical flow between the images, we have found that direct application of optical flow is not capable of estimating good correspondence when the underlying lip displacements between images are greater than 5 pixels.

It is possible to use *flow concatenation* to overcome this problem. Since the original corpus is digitized at 29.97 fps, there are many intermediate frames that lie between

the chosen prototypes. A series of consecutive optical flow vectors between each intermediate image and its successor may be computed and *concatenated* into one large flow vector that defines the global transformation between the chosen prototypes (see Appendix A for details on flow concatenation).

Typically, however, prototype images are very far apart in the recorded visual corpus, so it is not practical to compute concatenated optical flow between them. The repeated concatenation that would be involved across the hundreds or thousands of intermediate frames leads to a considerably degraded final flow.

To compute good correspondence between prototypes, a method is needed to figure out how to compute the path from the reference example I_1 to the chosen image prototypes I_i without repeated concatenation over hundreds or thousands of intermediates frames. We accomplish this by constructing the *corpus graph* representation of the corpus: A corpus graph is an S-by-S sparse adjacency graph matrix in which each frame in the corpus is represented as a node in a graph connected to k nearest images. The k nearest images are chosen using the *k-nearest neighbors* algorithm [6], and the distance metric used is the Mahalanobis distance in Equation 6.2 applied to the PCA parameters p. Thus, an image is connected in the graph to the k other images that look most similar to it. The edge-weight between a frame and its neighbor is the value of the Mahalanobis distance. We set $k = 20$ in this work.

After the corpus graph is computed, the *Dijkstra* shortest path algorithm [18] [40] is used to compute the shortest path between the reference example I_1 and the other chosen image prototypes I_i. Each shortest path produced by the Dijkstra algorithm is a list of images from the corpus that cumulatively represent the shortest deformation path from I_1 to I_i as measured by the Mahalanobis distance. Concatenated flow from I_1 to I_i is then computed along the intermediate images produced by the Dijkstra algorithm. Since there are 46 images, $N = 46$ correspondences $\{C_i\}_{i=1}^{N}$ are computed in this fashion from the reference image I_1 to the other image prototypes $\{I_i\}_{i=1}^{N}$.
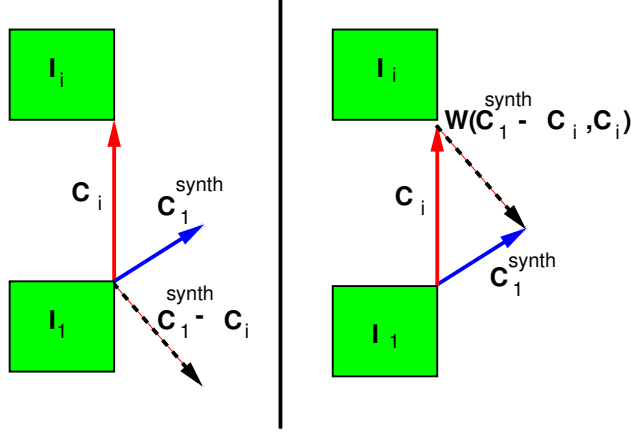
Figure 6-2: The flow reorientation process: First, C_i is subtracted from the synthesized flow C_1^{synth} . Second, this flow vector is itself forward warped along C_i.

6.3 Synthesis

The goal of synthesis is to map from the multidimensional parameter space (α, β) to an image which lies at that position in MMM space. Since there are 46 correspondences, α is a 46-dimensional parameter vector that controls mouth shape. Similarly, since there are 46 image prototypes, β is a 46-dimensional parameter vector that controls mouth texture. The total dimensionality of (α, β) is 92.

Synthesis first proceeds by synthesizing a new correspondence C^{synth} using *linear combination* of the prototype flows C_i:

$$C_1^{synth} = \sum_{i=1}^{N} \alpha_i C_i. \tag{6.3}$$

The subscript 1 in Equation 6.3 above is used to emphasize that C_1^{synth} originates from the reference image I_1, since all the prototype flows are taken with I_1 as reference.

Forward warping may be used to push the pixels of the reference image I_1 along the synthesized correspondence vector C_1^{synth}. Notationally, we denote the forward warping operation as an operator $\mathbf{W}(I, C)$ that operates on an image I and a correspondence map C (see Appendix B for details on forward warping).

However, a single forward warp will not utilize the image texture from *all* the examples. In order to take into account all image texture, a *correspondence re-*

30

orientation procedure first described in [5] is adopted that re-orients the synthesized correspondence vector C_1^{synth} so that it originates from each of the other example images I_i. Reorientation of the synthesized flow C_1^{synth} proceeds in two steps, shown figuratively in Figure 6-2. First, C_i is subtracted from the synthesized flow C_1^{synth} to yield a flow that contains the correct flow geometry, but which originates from the reference example I_1 rather than the desired example image I_i. Secondly, to move the flow into the correct reference frame, this flow vector is *itself warped* along C_i. The entire re-orientation process may be denoted as follows:

$$C_i^{synth} = \mathbf{W}(C_1^{synth} - C_i, C_i). \tag{6.4}$$

Re-orientation is performed for all examples in the example set.

The third step in synthesis is to warp the prototype images I_i along the re-oriented flows C_i^{synth} to generate a set of N warped image textures I_i^{warped}:

$$I_i^{warped} = \mathbf{W}(I_i, C_i^{synth}). \tag{6.5}$$

The fourth and final step is to blend the warped images I_i^{warped} using the β parameters to yield the final morphed image:

$$I^{morph} = \sum_{i=1}^{N} \beta_i I_i^{warped}. \tag{6.6}$$

Combining Equations 6.3 through 6.6 together, our MMM synthesis may be written as follows:

$$I^{morph}(\alpha, \beta) = \sum_{i=1}^{N} \beta_i \mathbf{W}(I_i, \mathbf{W}(\sum_{j=1}^{N} \alpha_j C_j - C_i, C_i)). \tag{6.7}$$

Empirically we have found that the MMM synthesis technique is capable of surprisingly realistic re-synthesis of lips, teeth, and tongue. However, the blending of multiple images in the MMM for synthesis tends to blur out some of the finer details in the teeth and tongue (See Appendix C for a discussion of synthesis blur). Shown in Figure 6-3 are some of the synthetic images produced by our system, along with

31

Figure 6-3: Top: Original images from our corpus. Bottom: Corresponding synthetic images generated by our system.

their real counterparts for comparison.

6.4 Analysis

The goal of analysis is to *project* the entire recorded corpus $\{I_j\}_{j=1}^{S}$ onto the constructed MMM, and produce a time series of $(\alpha_j, \beta_j)_{j=1}^{S}$ parameters that represent trajectories of the original mouth motion in MMM space.

One possible approach for analysis of images is to perform *analysis-by-synthesis*: In this approach, used in various forms in [25] [9], the synthesis algorithm is used to synthesize an image $I^{synth}(\alpha, \beta)$, which is then compared to the novel image using an error metric (ie, the L2 norm). Gradient-descent is then usually performed to change the parameters in order to minimize the error, and the synthesis process is repeated. The search ends when a local minimum is achieved. Analysis-by-synthesis, however, is very slow in the case when a large number of images are involved.

In this work we choose another method that is capable of extracting parameters (α, β) in one iteration. In addition to the image I^{novel} to be analyzed, the method requires that the correspondence C^{novel} from the reference image I_1 in the MMM to the novel image I^{novel} be computed beforehand. In our case, most of the novel imagery to be analyzed will be from the recorded video corpus itself, so we employ the Dijkstra approach discussed in Section 6.2.3 to compute good quality correspondences between the reference image I_1 and I_{novel}.

Given a novel image I^{novel} and its associated correspondence C^{novel}, the first step

32
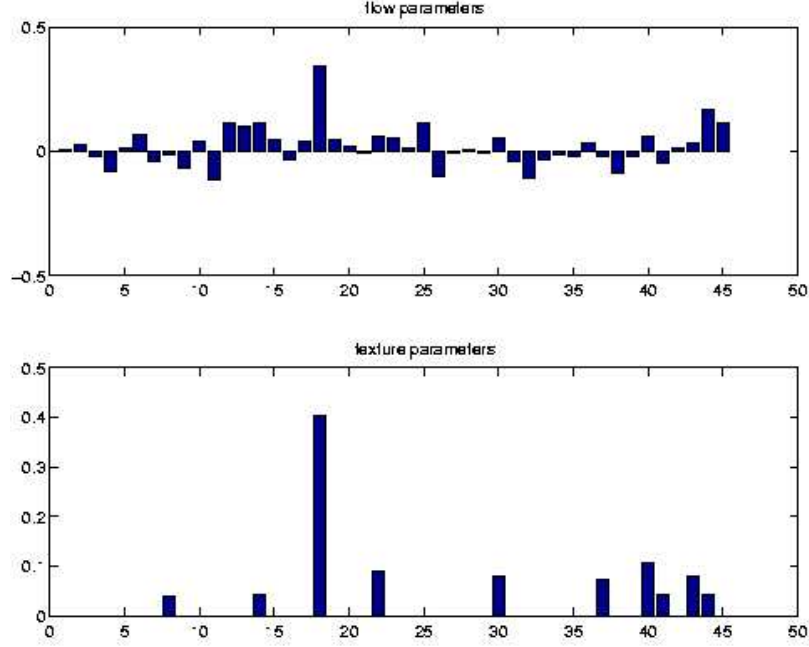
Figure 6-4: Top: Analyzed α_i flow parameters computed for one image. Bottom: The corresponding analyzed β_i texture parameters computed for the same image. The β_i texture parameters are typically zero for all but a few image prototypes.

of the analysis algorithm is to estimate the parameters α which minimize

$$\|C^{novel} - \sum_{i=1}^{N} \alpha_i C_i\|. \tag{6.8}$$

This is solved using the pseudo-inverse:

$$\alpha = \left(C^T C\right)^{-1} C^T C^{novel} \tag{6.9}$$

where C above is a matrix containing all the prototype correspondences $\{C_i\}_{i=1}^{N}$.

After the parameters α are estimated, N image warps are synthesized in the same manner as described in Section 6.3 using flow-reorientation and warping:

$$I_i^{warp} = \mathbf{W}\left(I_i, \mathbf{W}\left(\sum_{i=1}^{N} \alpha_i C_i - C_i, C_i\right)\right). \tag{6.10}$$

The final step in analysis is to estimate the values of β as the values which minimize

33

$$\|I^{novel} - \sum_{i=1}^{N} \beta_i I_i^{warp}\| \quad \texttt{subject to}$$

$$\beta_i > 0 \;\; \forall i \;\; \texttt{and} \;\; \sum_{i=1}^{N} \beta_i = 1. \tag{6.11}$$

The non-negativity constraint above on the β_i parameters ensures that pixel values are not negated. The normalization constraint ensures that the β_i parameters are computed in a normalized manner for each frame, which prevents brightness flickering during synthesis. The form of the imposed constraints cause the computed β_i parameters to be *sparse* (see Figure 6-4), which enables efficient synthesis by requiring only a few image warps (instead of the complete set of 46 warps). Equation 6.11, which involves the minimization of a quadratic cost function subject to constraints, is solved using quadratic programming methods. In this work, we use the Matlab function `quadprog`.

Each utterance in the corpus is analyzed with respect to the 92-dimensional MMM created in Section 6.2, yielding a set of $z_t = (\alpha_t, \beta_t)$ parameters for each utterance. Analysis takes on the order of 15 seconds per frame on a circa 1998 450 MHz Pentium II machine. Shown in Figure 7-2 in solid blue are example analyzed trajectories for α_{12} and β_{28} computed for the word `tabloid`.

Chapter 7

Trajectory Synthesis

7.1 Overview

The goal of trajectory synthesis is to map from an input *phone stream* $\{P_t\}$ to a *trajectory* $y_t = (\alpha_t, \beta_t)$ of parameters in MMM space. After the parameters are synthesized, Equation 6.7 from Section 6.3 is used to create the final visual stream that represents the talking face.

The phone stream is a stream of phonemes $\{P_t\}$ representing that phonetic transcription of the utterance. For example, the word one may be represented by a phone stream $\{P_t\}_{t=1}^{15} = (\backslash w\backslash, \backslash w\backslash, \backslash w\backslash, \backslash w\backslash, \backslash uh\backslash, \backslash uh\backslash, \backslash uh\backslash, \backslash uh\backslash, \backslash uh\backslash, \backslash uh\backslash, \backslash n\backslash, \backslash n\backslash, \backslash n\backslash, \backslash n\backslash, \backslash n\backslash)$. Each element in the phone stream represents one image frame. We define T to be the length of the entire utterance in frames.

Since the audio is aligned, it is possible to examine all the flow and texture parameters for any particular phoneme. Shown in Figure 7-1 are histograms for the α_1 parameter for the $\backslash w\backslash$, $\backslash m\backslash$, $\backslash aa\backslash$ and $\backslash ow\backslash$ phones. Evaluation of the analyzed parameters from the corpus reveals that parameters representing the same phoneme tend to *cluster* in MMM space. We represent each phoneme p mathematically as a multidimensional Gaussian with mean μ_p and diagonal covariance Σ_p. Separate means and covariances are estimated for the flow and texture parameters [1].

---

[1]Technically, since the texture parameters are non-negative, they are best modeled using Gamma distributions not Gaussians. In that case, Equation 7.1 needs to be re-written for Gamma distribu-
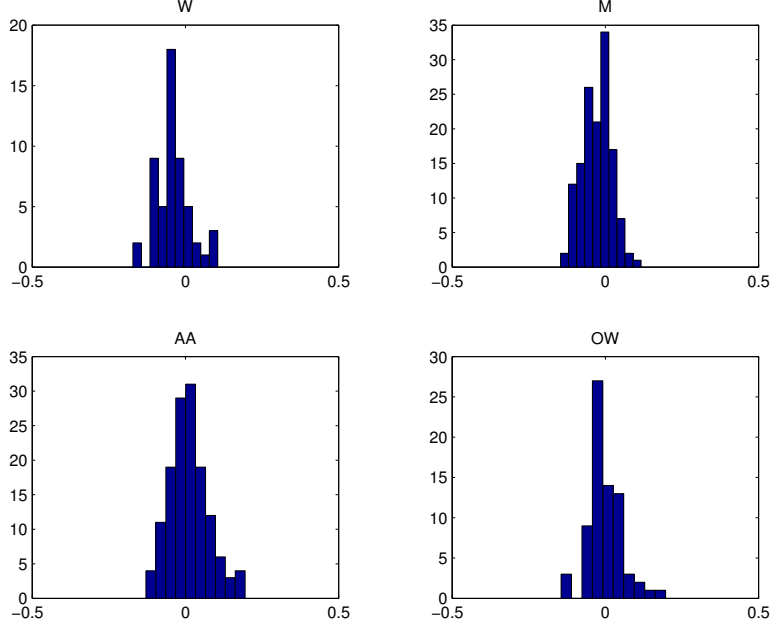
Figure 7-1: Histograms for the α_1 parameter for the \w\, \m\, \aa\ and \ow\ phones.

The trajectory synthesis problem is framed mathematically as a *regularization problem* [21] [42]. The goal is to synthesize a trajectory y which minimizes an objective function E consisting of a *target term* and a *smoothness term*:

$$E = \underbrace{(y - \mu)^T D^T \Sigma^{-1} D (y - \mu)}_{target\ term} + \lambda \underbrace{y^T W^T W y}_{smoothness}. \tag{7.1}$$

The desired trajectory y is a vertical concatenation of the individual $y_t = \alpha_t$ terms at each time step (or $y_t = \beta_t$, since we treat flow and texture parameters separately):

$$y = \begin{bmatrix} y_t \\ \vdots \\ y_T \end{bmatrix} \tag{7.2}$$

The target term consists of the relevant means μ and covariances Σ constructed from the phone stream:

tions. In practice, however, we have found Gaussians to work well enough for texture parameters.

$$\mu = \begin{bmatrix} \mu_{P_t} \\ \vdots \\ \mu_{P_T} \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{P_t} & & \\ & \ddots & \\ & & \Sigma_{P_T} \end{bmatrix} \tag{7.3}$$

The matrix D is a duration-weighting matrix which emphasizes the shorter phonemes and de-emphasizes the longer ones, so that the objective function is not heavily skewed by the phonemes of longer duration:

$$D = \begin{bmatrix} \sqrt{I - \frac{D_{P_1}}{T}} & & & \\ & \sqrt{I - \frac{D_{P_2}}{T}} & & \\ & & \ddots & \\ & & & \sqrt{I - \frac{D_{P_T}}{T}} \end{bmatrix} \tag{7.4}$$

One possible smoothness term consists of the first order difference operator:

$$W = \begin{bmatrix} -I & I & & \\ & -I & I & \\ & & \ddots & \\ & & & -I & I \end{bmatrix} \tag{7.5}$$

Higher orders of smoothness are formed by repeatedly multiplying W with itself: second order $W^T W^T W W$, third order $W^T W^T W^T W W W$, and so on.

Finally, the regularizer λ determines the trade-off between both terms.

Taking the derivative of Equation 7.1 and minimizing yields the following equation for synthesis:

$$(D^T \Sigma^{-1} D + \lambda W^T W) y = D^T \Sigma^{-1} D \mu. \tag{7.6}$$

Given known means μ, covariances Σ, and regularizer λ, synthesis is simply a matter of plugging them into Equation 7.6 and solving for y using Gaussian elimination. This is done separately for the flow and the texture parameters. In our experiments a regularizer of degree four yielding *multivariate additive septic splines* [42] gave satisfactory results (see next subsection).

Coarticulation effects in our system are modeled via the magnitude of the variance Σ_P for each phoneme. Small variance means the trajectory *must* pass through that region in phoneme space, and hence neighboring phonemes have little coarticulatory effect. On the other hand, large variance means the trajectory has a lot of flexibility in choosing a path through a particular phonetic region, and hence it may choose to pass through regions which are closer to a phoneme's neighbors. The phoneme will thus experience large coarticulatory effects.

There is no explicit model of phonetic dynamics in our system. Instead, phonetic dynamics emerge implicitly through the interplay between the magnitude of the variance Σ_P for each phoneme (which determines the phoneme's "spatial" extent), and the input phone stream (which determines the duration in time of each phoneme). Equation 7.1 then determines the speed through a phonetic region in a manner which balances nearness to the phoneme with smoothness of the overall trajectory. In general, we find the trajectories speed up in regions of small duration and small variance (ie plosives), while they slow down in regions of large duration and large variance (ie silences).

7.2   Training

The means μ_p and covariances Σ_p for each phone p are initialized directly from the data using sample means and covariances. However, the sample estimates tend to average out the mouth movement so that it looks under-articulated. As a consequence, there is a need to adjust the means and variances to better reflect the training data.

Gradient descent learning [6] is employed to adjust the mean and covariances. First, the Euclidean error metric is chosen to represent the error between the original utterance z and the synthetic utterance y:

$$E = (z - y)^T (z - y). \tag{7.7}$$

The parameters $\{\mu_p, \Sigma_p\}$ need to be changed to minimize this objective function E. The chain rule may be used to derive the relationship between E and the parameters:
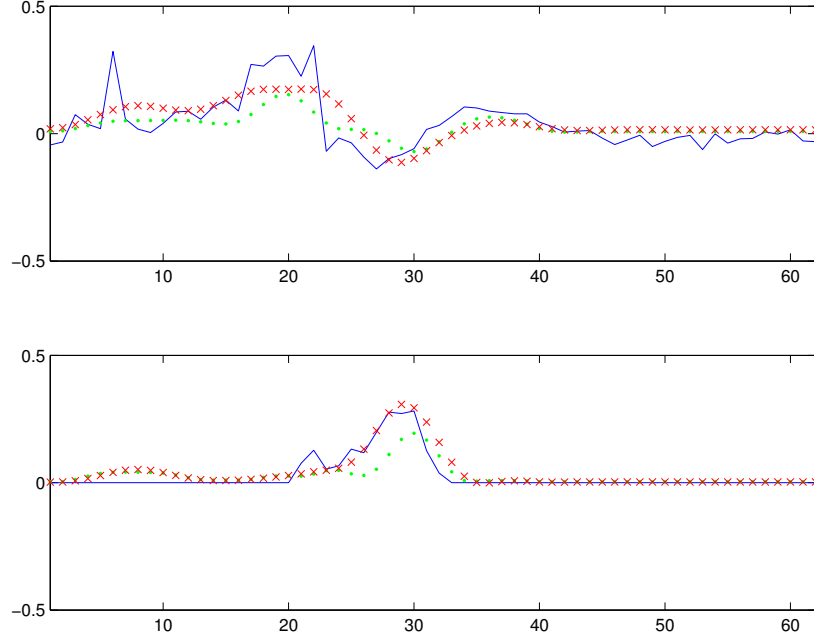
Figure 7-2: Top: The analyzed trajectory for α_{12} (in solid blue), compared with the synthesized trajectory for α_{12} before training (in green dots) and after training (in red crosses). Bottom: Same as above, but the trajectory is for β_{28}. Both trajectories are from the word `tabloid`.

$$\frac{\partial E}{\partial \mu_i} = \left(\frac{\partial E}{\partial y}\right)^T \left(\frac{\partial y}{\partial \mu_i}\right) \tag{7.8}$$

$$\frac{\partial E}{\partial \sigma_{ij}} = \left(\frac{\partial E}{\partial y}\right)^T \left(\frac{\partial y}{\partial \sigma_{ij}}\right). \tag{7.9}$$

$\frac{\partial E}{\partial y}$ may be obtained from Equation 7.7:

$$\frac{\partial E}{\partial y} = -2(z - y). \tag{7.10}$$

Since y is defined according to Equation 7.6, we can take its derivative to compute $\frac{\partial y}{\partial \mu_i}$ and $\frac{\partial y}{\partial \sigma_{ij}}$:

$$(D^T \Sigma^{-1} D + \lambda W^T W)\frac{\partial y}{\partial \mu_i} = D^T \Sigma^{-1} D \frac{\partial \mu}{\partial \mu_i} \tag{7.11}$$

$$(D^T \Sigma^{-1} D + \lambda W^T W) \frac{\partial y}{\partial \sigma_{ij}} =$$

$$2 D^T \Sigma^{-1} \frac{\partial \Sigma}{\partial \sigma_{ij}} \Sigma^{-1} D(y - \mu). \qquad (7.12)$$

Finally, gradient descent is performed by changing the previous values of the parameters according to the computed gradient:

$$\mu^{new} = \mu^{old} - \eta \frac{\partial E}{\partial \mu} \qquad (7.13)$$

$$\Sigma^{new} = \Sigma^{old} - \eta \frac{\partial E}{\partial \Sigma}. \qquad (7.14)$$

Cross-validation sessions were performed to evaluate the appropriate value of λ and the correct level of smoothness W to use. The learning rate η was set to 0.00001 for all trials, and 10 iterations performed. Comparison between batch and online updates indicated that online updates perform better, so this method was used throughout training. Testing was performed on a set composed of 1-syllable words, 2-syllable words, and sentences *not contained in the training set.* The Euclidean norm between the synthesized trajectories and the original trajectories was used to measure error. The results showed that the optimal smoothness operator is *fourth order* and the optimal regularizer is $\lambda = 1000$. Figure 7-2 depicts synthesized trajectories for the α_{12} and β_{28} parameters before training (in green dots) and after training (in red crosses) for these optimal values of W and λ.

Chapter 8

Post-Processing

Due to the head and eye normalization that was performed during the pre-processing stage, the final animations generated by our system exhibit movement only in the mouth region. This leads to an unnerving "zombie"-like quality to the final animations. To address this, we composite the synthesized mouth onto a background sequence which contains natural head and eye movement.

8.1 Adding Noise

The first step in the composition process is to add Gaussian noise to the synthesized images to regain the camera image sensing noise that is lost as a result of blending multiple image prototypes in the MMM. We estimate means and variances for this noise by computing differences between original images and images synthesized by our system, and averaging over 200 images. Shown in Figure 8-1 at the top are estimates for this noise for the R channel. Generally this process yields unacceptably high noise variances around the mouth region due to synthesis mismatch, so we replace these estimates with values from the cheek region.
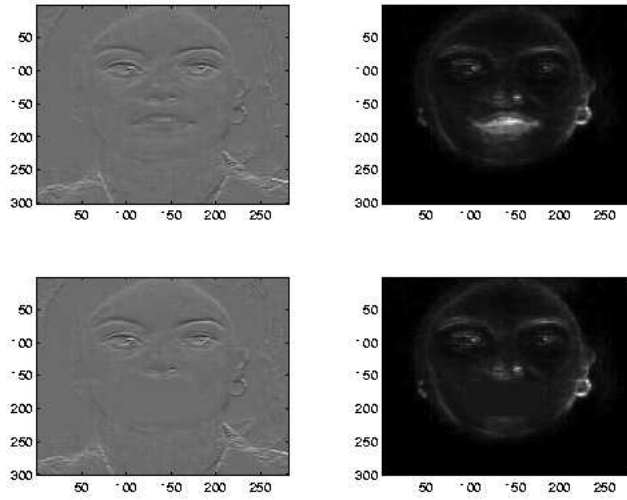
Figure 8-1: Top: Estimated mean and standard deviations for the image error between original and synthetic images. The values are unacceptably high around the mouth region, leading to high flicker around the mouth region if the noise is sampled. Bottom: The error values after the area around the mouth region is replaced with more acceptable values from the cheek region.

8.2   Compositing onto a Background Sequence

After noise is added, the synthesized sequences are composited onto the chosen background sequence with the help of the masks shown in Figure 5-1. The head mask is first forward warped using optical flow to fit across the head of each image of the background sequence. Next, optical flow is computed between each background image and its corresponding synthetic image. The synthetic image and the mouth mask from Figure 5-1 are then perspective-warped back onto the background image. The perspective warp is estimated *using only the flow vectors lying within the background head mask*. The final composite is made by pasting the warped mouth onto the background image using the warped mouth mask. The mouth mask is smoothed at the edges to perform a seamless blend between the background image and the synthesized mouth. The compositing process is depicted in Figure 8-2.

42

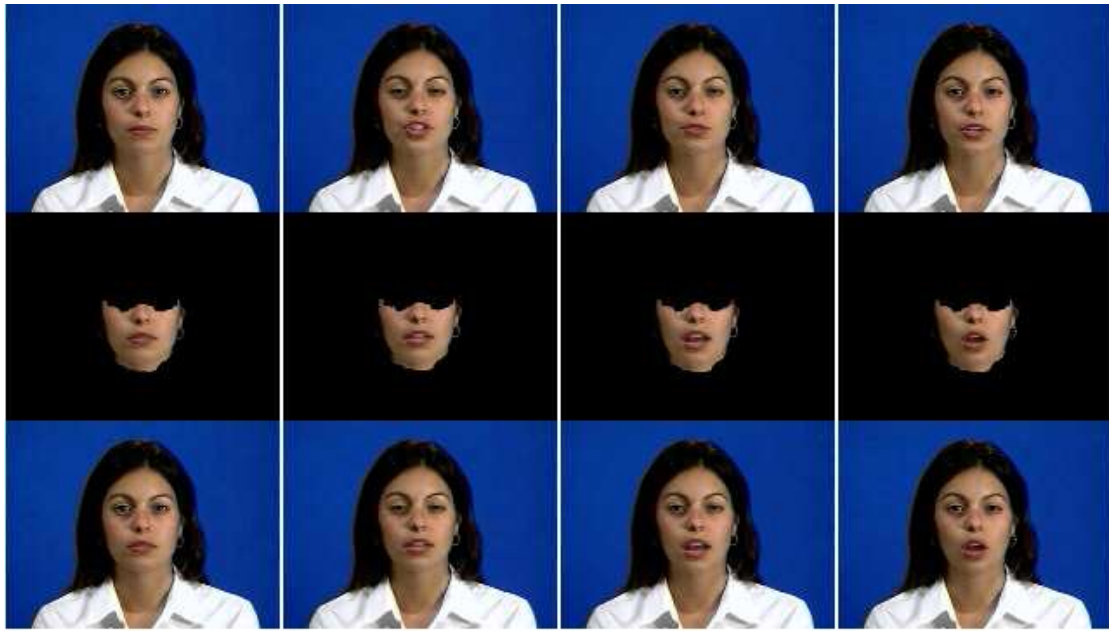Figure 8-2: The background compositing process: Top: A background sequence with natural head and eye movement. Middle: A sequence generated from our system, with the desired mouth movement and appropriate masking. Bottom: The final composited sequence with the desired mouth movement, but with the natural head and eye movements of the background sequence. The masks from Figure 5-1 are used to guide the compositing process.

Chapter 9

Computational Issues

To use our system, an animator first provides phonetically annotated audio. The annotation may be done automatically [24], semi-automatically using a text transcript [24], or manually [39].

Trajectory synthesis is performed by Equation 7.6 using the trained phonetic models. This is done separately for the flow and the texture parameters. After the parameters are synthesized, Equation 6.7 from Section 6.3 is used to create the visual stream with the desired mouth movement. Typically only the image prototypes I_i which are associated with top 10 values of β_i are warped, which yields a considerable savings in computation time. MMM synthesis takes on the order of about 7 seconds per frame for an image resolution of 624x472. The background compositing process adds on a few extra seconds of processing time. All times are computed on a 450 MHz Pentium II.

Chapter 10

Evaluation

We have synthesized numerous examples using our system, spanning the entire range of 1-syllable words, 2-syllable words, short sentences, and long sentences. In addition, we have synthesized songs and foreign speech examples.

Experimentally we have found that reducing the number of prototypes below 30 degrades the quality of the final animations. An open question is whether *increasing* the number of prototypes significantly beyond 46 will lead to even higher levels of videorealism.

In terms of corpus size, it is possible to optimize the spoken corpus so that several words alone elicit the 46 prototypes. This would reduce the duration of the corpus from 15 minutes to a few seconds. However, this would degrade the quality of the correspondences computed by the Dijkstra algorithm. In addition, the phonetic training performed by our trajectory synthesis module would degrade as well. Further systematic experiments need to be made in order to evaluate how final performance changes with the size of the corpus.

We evaluated our results by performing three different visual "Turing tests" to see whether human subjects can distinguish between real sequences and synthetic ones. In the first experiment ("single presentation"), subjects were asked to view *one* visual sequence at a time, and identify whether it is real or synthetic. In a similar second experiment ("fast single presentation"), the subjects were asked to make the judgments in a fast manner while the utterances were being presented without pauses

Experiment	# subjects	% correct	t	p<
Single pres.	22	54.3%	1.243	0.3
Fast single pres.	21	52.1%	0.619	0.5
Double pres.	22	46.6%	-0.75	0.5

Table 10.1: Levels of correct identification of real and synthetic sequences. "t" represents the value from a standard t-test with significance level indicated in the "p<" column.

in between. In a third experiment ("double presentation"), the subjects were asked to view pairs of the same utterance, where one item in the pair is real and the other is synthetic (but randomly ordered). The subjects in this experiment were asked to identify which utterance in the pair is real, and which is synthetic. 16 or 18 utterances were presented to each subject, with half being real and half being synthetic. As seen from Table 10.1, performance in all three experiments was close to chance level (50%) and not significantly different from it.

Finally, we also evaluated our system by performing intelligibility tests in which subjects were asked to lip read a set of natural and synthetic utterances.

Details on all experiments are forthcoming in a separate article.

Chapter 11

Further Work

The main limitation of our technique is the difficulty of re-compositing synthesized mouth sequences into background sequences which involve 1) large changes in head pose, 2) changes in lighting conditions, and 3) changes in viewpoint. All these limitations can be alleviated by extending our approach from 2D to 3D. It is possible to envision a real-time 3D scanner that is capable of recording a 3D video corpus of speech. Alternatively, techniques such as those presented in [22] [35] [9] can be used to map a 2D video corpus into 3D.

The geodesic trajectory synthesis equations described by Brand et al. [10] [11] are analogous (and more sophisticated) than the trajectory synthesis techniques we use (Equations 7.1 and 7.6). Although those equations require considerably more training data, it is possible they could lead to higher levels of videorealism.

Clearly the face is used as a conduit to transmit emotion, so one possible avenue to explore is the synthesis of speech under various emotional states. It is possible to record various corpora under different emotional states and create MMMs for each state. During synthesis, the appropriate MMM is selected. An open question to explore is *emotional dynamics*: how does one transition from a happy MMM to a sad MMM? Additionally, there is also a need to learn generative models of head movement and eye movement tailored for the type of speech being synthesized.

Appendix A

Appendix: Flow Concatenation

Given a series of consecutive images $I_0, I_1, \ldots I_n$, we would like to construct the correspondence map $C_{0(n)}$ relating I_0 to I_n. We focus on the case of the 3 images I_{i-1}, I_i, I_{i+1} since the concatenation algorithm is simply an iterative application of this 3-frame base case. Optical flow is first computed between the consecutive frames to yield $C_{(i-1)i}, C_{i(i+1)}$. Note that it is not correct to construct $C_{(i-1)(i+1)}$ as the simple addition of $C_{(i-1)i} + C_{i(i+1)}$ because the two flow fields are with respect to two different reference images. Vector addition needs to be performed with respect to a common origin.

Our concatenation thus proceeds in two steps: to place all vector fields in the same reference frame, the *correspondence map $C_{i(i+1)}$ itself* is warped *backwards* [45] along $C_{(i-1)i}$ to create $C_{i(i+1)}^{warped}$. Now $C_{i(i+1)}^{warped}$ and $C_{(i-1)i}$ are both added to produce an approximation to the desired concatenated correspondence:

$$C_{(i-1)(i+1)} = C_{(i-1)i} + C_{i(i+1)}^{warped}. \tag{A.1}$$

A procedural version of our backwarp warp is shown in figure A-1. BILINEAR refers to bilinear interpolation of the 4 pixel values closest to the point (x,y).

```
for j = 0...height,
  for i = 0...width,
    x = i + dx(i,j);
    y = j + dy(i,j);
    I^{warped}(i,j) = BILINEAR (I, x, y);
```

Figure A-1: **BACKWARD WARP** algorithm

Appendix B

Appendix: Forward Warping

Forward warping may be viewed as "pushing" the pixels of an image I along the computed flow vectors C. We denote the forward warping operation as an operator $\mathbf{W}(I, C)$ that operates on an image I and a correspondence map C, producing a warped image I^{warped} as final output. A procedural version of our forward warp is shown in Figure B-1.

It is also possible to forward warp a correspondence map C' along another correspondence C, which we denote as $\mathbf{W}(C', C)$. In this scenario, the x and y components of $C'(\mathbf{p}) = \{d'_x(\mathbf{p}), d'_y(\mathbf{p})\}$ are treated as separate images, and warped individually along C: $\mathbf{W}(dx', C)$ and $\mathbf{W}(dy', C)$.

```
for j = 0...height,
   for i = 0...width,
      x = ROUND (i + αdx(i,j) );
      y = ROUND (j + αdy(i,j) );
      if (x,y) are within the image
         I^warped(x,y) = I(i,j);
```

Figure B-1: **FORWARD WARP** algorithm

Appendix C

Appendix: Hole-Filling

Forward warping produces *black holes* which occur in cases where a destination pixel was not filled in with any source pixel value. This occurs due to inherent nonzero divergence in the optical flow, particularly around the region where the mouth is expanding. To remedy this, a hole-filling algorithm [15] was adopted which pre-fills a destination image with a special reserved background color. After warping, the destination image is traversed in rasterized order and the holes are filled in by interpolating linearly between their non-hole endpoints.

In the context of our synthesis algorithm in Section 6.3, hole-filling can be performed *before blending*, or *after blending*. Throughout this paper, we assume hole-filling is performed before blending, which allows us to subsume the hole-filling procedure into our forward warp operator $\mathbf{W}$ and simplify our notation. Consequently (as in Equation 6.6), the blending operation becomes a simple linear combination of the hole-filled warped intermediates I_i^{warped}.

In practice, however, we perform hole-filling *after blending*, which reduces the size of the holes that need to be filled, and leads to a considerable reduction in synthesis blur. Post-blending hole-filling requires a more complex blending algorithm than as noted in Equation 6.6 because the blending algorithm now needs to keep track of holes and non-holes in the warped intermediate images I_i^{warped}:

$$I^{morph}(x, y) = \frac{\sum_{I_i^{warped}(x,y) \neq hole} \beta_i I_i^{warped}(x, y)}{\sum_{I_i^{warped}(x,y) \neq hole} \beta_i} \tag{C.1}$$

Typically an accumulator array is used to keep track of the denominator term in Equation C.1 above. The synthesized mouth images shown in Figure 6-3 were generated using post-blending hole-filling.

Bibliography

[1] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[2] T. Beier and S. Neely. Feature-based image metamorphosis. In *Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, volume 26(2), pages 35–42, Chicago, IL, 1992. ACM.

[3] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proceedings of the European Conference on Computer Vision*, pages 237–252, Santa Margherita Ligure, Italy, 1992.

[4] D. Beymer and T. Poggio. Image representations for visual learning. *Science*, 272:1905–1909, 1996.

[5] D. Beymer, A. Shashua, and T. Poggio. Example based image analysis and synthesis. Technical Report 1431, MIT AI Lab, 1993.

[6] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[7] A. Black and P. Taylor. *The Festival Speech Synthesis System*. University of Edinburgh, 1997.

[8] M. Black, D. Fleet, and Y. Yacoob. Robustly estimating changes in image appearance. *Computer Vision and Image Understanding, Special Issue on Robust Statistical Techniques in Image Understanding*, pages 8–31, 2000.

[9] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In Alyn Rockwood, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 187–194, Los Angeles, 1999. ACM, ACM Press / ACM SIGGRAPH.

[10] M. Brand. Voice puppetry. In Alyn Rockwood, editor, *Proceedings of SIG-GRAPH 1999*, Computer Graphics Proceedings, Annual Conference Series, pages 21–28, Los Angeles, 1999. ACM, ACM Press / ACM SIGGRAPH.

[11] M. Brand and A. Hertzmann. Style machines. In Kurt Akeley, editor, *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 183–192. ACM, ACM Press / ACM SIGGRAPH, 2000.

[12] C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual speech with audio. In *Proceedings of SIGGRAPH 1997*, Computer Graphics Proceedings, Annual Conference Series, pages 353–360, Los Angeles, CA, August 1997. ACM, ACM Press / ACM SIGGRAPH.

[13] N.M. Brooke and S.D. Scott. Computer graphics animations of talking faces based on stochastic models. In *Intl. Symposium on Speech, Image Processing, and Neural Networks*, Hong Kong, April 1994.

[14] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. on Communications*, COM-31(4):532–540, April 1983.

[15] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proceedings of SIGGRAPH 1993*, Computer Graphics Proceedings, Annual Conference Series, pages 279–288, Anaheim, CA, August 1993. ACM, ACM Press / ACM SIGGRAPH.

[16] M. M. Cohen and D. W. Massaro. Modeling coarticulation in synthetic visual speech. In N. M. Thalmann and D. Thalmann, editors, *Models and Techniques in Computer Animation*, pages 139–156. Springer-Verlag, Tokyo, 1993.

[17] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *Proceedings of the European Conference on Computer Vision*, Freiburg, Germany, 1998.

[18] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press and McGraw-Hill Book Company, 1989.

[19] E. Cosatto and H. Graf. Sample-based synthesis of photorealistic talking heads. In *Proceedings of Computer Animation '98*, pages 103–110, Philadelphia, Pennsylvania, 1998.

[20] T. Ezzat and T. Poggio. Visual speech synthesis by morphing visemes. *International Journal of Computer Vision*, 38:45–57, 2000.

[21] F. Girosi, M. Jones, and T. Poggio. Priors, stabilizers, and basis functions: From regularization to radial, tensor, and additive splines. Technical Report 1430, MIT AI Lab, June 1993.

[22] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *Proceedings of SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, pages 55–66, Orlando, FL, 1998. ACM, ACM Press / ACM SIGGRAPH.

[23] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[24] X. Huang, F. Alleva, H.-W. Hon, M.-Y. Hwang, K.-F. Lee, and R. Rosenfeld. The SPHINX-II speech recognition system: an overview (http://sourceforge.net/projects/cmusphinx/). *Computer Speech and Language*, 7(2):137–148, 1993.

[25] M. Jones and T. Poggio. Multidimensional morphable models: A framework for representing and maching object classes. In *Proceedings of the International Conference on Computer Vision*, Bombay, India, 1998.

[26] A. Lanitis, C.J. Taylor, and T.F. Cootes. A unified approach to coding and inter- preting face images. In *Proceedings of the International Conference on Computer Vision*, pages 368–373, Cambridge, MA, June 1995.

[27] S. Y. Lee, K. Y. Chwa, S. Y. Shin, and G. Wolberg. Image metemorphosis using snakes and free-form deformations. In *Proceedings of SIGGRAPH 1995*, volume 29 of *Computer Graphics Proceedings, Annual Conference Series*, pages 439–448. ACM, ACM Press / ACM SIGGRAPH, 1995.

[28] S. Y. Lee, G. Wolberg, and S. Y. Shin. Polymorph: An algorithm for morphing among multiple images. *IEEE Computer Graphics Applications*, 18:58–71, 1998.

[29] Y. Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *Proceedings of SIGGRAPH 1995*, Computer Graphics Proceedings, Annual Conference Series, pages 55–62, Los Angeles, California, August 1995. ACM, ACM Press / ACM SIGGRAPH.

[30] B. LeGoff and C. Benoit. A text-to-audiovisual-speech synthesizer for french. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Philadelphia, USA, October 1996.

[31] T. Masuko, T. Kobayashi, M. Tamura, J. Masubuchi, and K. Tokuda. Text-to- visual speech synthesis based on parameter generation from hmm. In *ICASSP*, 1998.

[32] E. Moulines and F. Charpentier. Pitch-synchronous waveform processing tech- niques for text-to-speech synthesis using diphones. *Speech Communication*, 9:453–467, 1990.

[33] F. I. Parke. *A parametric model of human faces*. PhD thesis, University of Utah, 1974.

[34] A. Pearce, B. Wyvill, G. Wyvill, and D. Hill. Speech and expression: A computer solution to face animation. In *Graphics Interface*, 1986.

[35] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. Salesin. Synthesizing realistic facial expressions from photographs. In *Proceedings of SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, pages 75–84, Orlando, FL, 1998. ACM, ACM Press / ACM SIGGRAPH.

[36] T. Poggio and T. Vetter. Recognition and structure from one 2D model view: observations on prototypes, object classes and symmetries. Technical Report 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[37] S. Roweis. EM algorithms for PCA and SPCA. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[38] K.C. Scott, D.S. Kagels, S.H. Watson, H. Rom, J.R. Wright, M. Lee, and K.J. Hussey. Synthesis of speaker facial movement to match selected speech sequences. In *Proceedings of the Fifth Australian Conference on Speech Science and Technology*, volume 2, pages 620–625, December 1994.

[39] K. Sjlander and J. Beskow. Wavesurfer - an open source speech tool. In *Proc of ICSLP*, volume 4, pages 464–467, Beijing, 2000.

[40] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, Dec 2000.

[41] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999.

[42] G. Wahba. *Splines Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

[43] K. Waters. A muscle model for animating three-dimensional facial expressions. In *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, volume 21(4), pages 17–24. ACM, July 1987.

[44] S.H. Watson, J.R. Wright, K.C. Scott, D.S. Kagels, D. Freda, and K.J. Hussey. An advanced morphing algorithm for interpolating phoneme images to simulate speech. Jet Propulsion Laboratory, California Institute of Technology, 1997.

[45] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA., 1990.